== Paper summary ==

The chapter stresses out some of the problems present in current memory system, and tries to alleviate some of the issues by offering some interesting solutions: new memory architectures, new uses of emerging technologies that might replace or coexist among the current memory systems, methods of offering predictable performance and quality of service among the current memory system.

The paper identifies three current fronts on which the current system is becoming a bottleneck. First, the architecture front (many changes at the architecture level such as heterogeneous systems, different users/agents that are using the memory system and require a good response from the memory system - QoS). Second, the application front (applications are becoming more data hungry and need a lot of capacity and bandwidth to be supplied). Lastly, the technology front (the current DRAM system do not scale anymore, therefore new technology nodes must be used for a better scaling).

The paper tries to offer some solutions to alleviate some of the problems on all of the fronts, it does not go into too much detail. However, it paints an interesting picture where the memory system is the main actor. The methods used for improving the memory system have some interesting background ideas and really show performance improvements. The chapter opens this vast research field offering some possible solutions but also leaving the reader to find and come up with other solutions by himself.

== Strengths of paper and mechanisms (bullet points) ==

1. The chapter is well structured and presents the information in a well-thought flow.

2. The problems that are being tackled are important problems that are causing the memory system to be a bottleneck.

3. The idea of refreshing only some regions of the DRAM to reduce power consumption.

4. Improving DRAM bank parallelism by favoring the sub arrays. Depending on how the data is being placed and additional of circuitry may offer a high improvement for dram parallelism.

5. The idea of moving some of the computing operations to the DRAM side seems very appealing. DRAM becomes the active player in the system and it can overlap memory operations with actual computations of the compute nodes, thereby reducing some latencies.

6. The chapter also tackles the issue of heterogeneous memory systems that contain both DRAM and other type of NVM systems.

7. The MISE idea of creating a model based on the memory requests presents an interesting solution that offers QoS when running multiple applications.

== Weaknesses of paper and mechanisms (bullet points) ==

1. The main issue with the retention-time-aware memory controller is how to determine that time. Profiling the application once and creating a mechanism for prediction may seem an interesting solution. That will require an accurate prediction and understanding of the application.

2. For improving the DRAM parallelism and the reduction of the latencies, the memory controller must know exactly where and how to put the data so that the system can make use of its mechanism. How complex will the memory controller be?

3. Making the DRAM an active player in the system will determine the creation of coherence mechanisms between compute nodes and memory system. Methods for notifying the compute nodes are needed.

4. What elements will notify the compute nodes as fast as possible when DRAM performs computations on data, so the compute nodes don't have stale values for

computation?

5. In hybrid memory systems, how will reliability be guaranteed? If the NVM memory system fails, how will the data that was in computation be restored?

== Detailed comments (expand on the summary as necessary) ==

As stated in the introduction, the ideas and algorithms, that were offered, help solve some of the issues that the memory system faces nowadays. All the mechanisms present good improvements, but can we do better, that is the main question. All solutions may have some drawbacks, such as RAIDR which tends to reduce power consumption because of the increasing need of refresh rates. The issue is that the collection of the retention times is difficult. The mechanism of prediction must be well put, and also a mechanism for recovery to offer reliability.

== Ideas for improvement - Can you do better? ==

In all of the cases, an analysis on the software is always needed. Each application is going to be characterized by a compute phase, transition phase, and memory phase. If the analysis on the memory phase is well done and for each application it can be predicted where the data is being placed, how the data is being used, then the mechanism will have a better success. There always has to be a cooperation between hardware and software. Compilers must also take into account the underlying hardware subtleties so as to offer the best optimizations. All the decisions done at the lower level must also be exposed some how to the upper level of the compute stack.

== Lessons learned ==

To conclude, the chapter presented some interesting problems that need to be solved so as to have high performance, whether we are talking about total execution time or power consumption. The chapter presented some serious requirements that need to be satisfied such as: capacity, bandwidth, reliability, performance, cost and so on. Overall the chapter was an interesting reading that covered a lot of material and presented some interesting direction for future research work.